Canary SRAM Built in Self-test for SRAM Write V_{MIN} Tracking

Arijit Banerjee

University of Virginia, Charlottesville, VA 22904, USA; E-mail: ab9ca@virginia.edu

Abstract— **As we shrink down devices with technology scaling, process variation increases and it hinders SRAM V_{MIN} scaling. Using peripheral assists, we can further lower the V_{MIN} at the cost of energy and area. However, the SRAM V_{MIN} varies with voltage, temperature and operating frequency variations, and it is hard to determine in real time. Prior work shows theoretically that canary SRAMs using reverse assist can track SRAM write V_{MIN}. However, during manufacturing the canary SRAM can have defects that can result in an erroneous SRAM V_{MIN} tracking. In order to check whether to use canaries to track SRAM write V_{MIN}, we need to have a built in self-test hardware that can test if the canary SRAM with reverse assist has a distinct bit failure trend. In this paper, we propose the first built in self-test hardware for a 512b canary SRAM with reverse assists in a 130nm bulk technology. We show that this hardware can produce go-no-go testing results during production test run to decide whether to use the canary SRAM in SRAM V_{MIN} tracking. In addition, the hardware provides debug functionality to pinpoint the issues in the canary SRAM failure trend, point by point.**

Index Terms **— Canary SRAM BIST, CBIST, dynamic write V_{MIN} tracking, Wordline and bitline type reverse assist.**

I. Introduction

As we shrink down devices with technology scaling the process variation increases, and the minimum operating voltage (V_{MIN}) of SRAMs gets higher. In order to lower the SRAM V_{MIN} further, we can use peripheral assists [1] for SRAM read and write operations. However, these assists come with area and power overhead. On the other hand, in [2] authors show that SRAM write failures are more prone to occur than read failures in deep submicron technologies. Hence, SRAM write V_{MIN} can be a bottleneck in lowering the overall V_{MIN} of a system on chip (SoC). The benefits of operating a SoC at SRAM write V_{MIN} is to save on design margins and power. However, the SRAM write V_{MIN} changes with process, voltage, temperature, and frequency variations and it is hard to determine it in real time. Canary circuits have been proposed to track SRAM design metrics using closed loop solutions [3][4]. In [4], authors have proposed a detailed theory to track SRAM write V_{MIN} using canary SRAM and reverse assist. However, due to defects in fabrication process, the V_{MIN} tracking using the canary bit failure rate (BFR) can be erroneous. Hence, we need a way to check if the canary SRAM is at all usable for SRAM write V_{MIN} tracking after manufacturing. In this paper, we show a canary SRAM built in self-test (CBIST) that can check the BFR trends for the go-no-go production run as well provide debug information related to the BFR trend. We organize the rest of the paper into five sections. Section II briefly describes the concept of peripheral assists and reverse assists and fault modelling of canary SRAMs. In section III, we describe the expected outcomes of the project. Section IV describes the approach of this project using behavioral flowchart and block diagrams. Section V describes the results, and finally, we conclude in section VI.

II. Peripheral Assists, Reverse Assists And Fault Modelling Of Canary SRAMs

As discussed earlier, one of the ways to reduce SRAM dynamic write V_{MIN} is to use SRAM peripheral assists [1]. On the other hand, a reverse assist [4] weakens an SRAM read or write operation and makes it vulnerable to read or write failures. Hence, we can use reverse assists in a smaller canary SRAM [3][4].to make it more prone to failure and use it as a sensor to track the dynamic write V_{MIN} of a bigger SRAM. The behavior of the reverse assist causes increase in bit failures during writing "1," "0" or both, which we can model as transition faults. The canary BFR or transition faults should have a unique trend across the degree of reverse assist (settings) and it should be monotonically increasing up to the maximum value. However, due to process variation, this BFR trend can have an upper and lower bound trends. For an example in the fast-fast (FF) corner the canary BFR trend can be very steep and in the slow-fast (SF) or slow-slow (SS) corner the BFR trend can be very slowly varying with the reverse assist settings. Hence, we check the canary SRAM against this maximum BFR (MXBFR) and minimum BFR (MNBFR) trends during the production run and see if the calculated BFR is within these two bounds and it is monotonically increasing at the same time. Note that we assume throughout the paper that the canary SRAM is of 512 bit with column mux 4 configuration and it has inbuilt reverse assist in it. Hence, we have 16 words of 32-bit width. In addition, we assume that the reverse assist has maximum of eight settings.

III. Expected Outcomes

The outcome of this work is a synthesized CBIST netlist from Verilog behavioral description in a commercial 130nm technology. The CBIST should be able to test the canaries at speed to check if the calculated BFR trend for canaries across the reverse assist settings is within the MXBFR and MNBFR bounds and it is monotonically increasing. In addition, the

CBIST should have a single signal go-no-go output for the production run to provide point by point debug functionality through a status register (BSTAT) for the calculated BFR trends. In order to test the CBIST itself we plan to incorporate full-scan insertion in the CBIST itself. Furthermore, we plan to calculate the fault coverage of the full-scan DFT structure.

IV. Approach Of This Work

The design of the CBIST must include a way to generate the address, data, reverse assist settings and read write signals for controlling the canary SRAM. Fig 1 shows the algorithm for the CBIST finite state machine (FSM) (CFSM) for looping around the addresses, and reverse assist settings during the canary SRAM testing. Here, we first initialize the canary SRAM using the initialization word (INITW) for all the 16 words. Thereafter, we write the complement of the INITW (INITBW) at speed and read it in a relaxed clock cycle for all the 16 words to ensure that the canary read operation is correct and only influence is in the written data. This loop runs for all the RaSel values and it runs one more time using INITBW also, which is not shown in Fig 2.

We incorporate another algorithm to compute the BFR (BFRFSM), which is shown in Fig 2. The BFRFSM computes the number of write failures in the canary SRAM in there steps. First, it does a bitwise XOR operation of the canary data in and data out bus. Then it counts the number of '1's using a bit adder (Fig 2). Thereafter, the bit error accumulator accumulates the number of errors to get the total BFR rate. In addition, the BFRFSM algorithm checks if the calculated BFR is within the bounds of MXBFR and MNBFR and asserts '1' or '0' in the BSTAT register (Fig 2) to indicate a valid or erroneous output accordingly. We repeat this loop for all the RaSels and INITW and INITBW words (not shown in Fig 2).

The high-level architecture of the CBIST is shown in Fig 3 that resembles a simplified typical memory BIST [5]. It comprises of three sub-blocks as CFSM, BFRFSM, and BFR Table to MXBFR and MNBFR interface. We already discussed the algorithm of the CFSM and BFRFSM blocks. We define the whole MXBFR and MNBFR trend data using MNBFR-MNBFR Table (MXMNBT). The MXMNBT to MNBFR-MNBFR interface converts the MXMNBT data into MNBFR and MXBFR data by selecting the apt data entry using RaSel values accordingly (Fig 3).

Once we are done with the Verilog behavioral design, we did the pre-synthesis RTL verification of the CBIST using Synopsys's VCS simulator in DVE environment. The testbench for RTL verification of CBIST reads randomly generated transition faults for simulation based verification, which we generate using a Perl script. Thereafter, the testbench injects the faults in the whole canary SRAM model per reverse assist setting (RaSel). In addition, we pre-calculated the output signature of the canary BFR results in hexadecimal. Once the simulation is done, it shows the calculated BFR table as the calculated hex signature, which should match the pre-calculated signature. After we are done

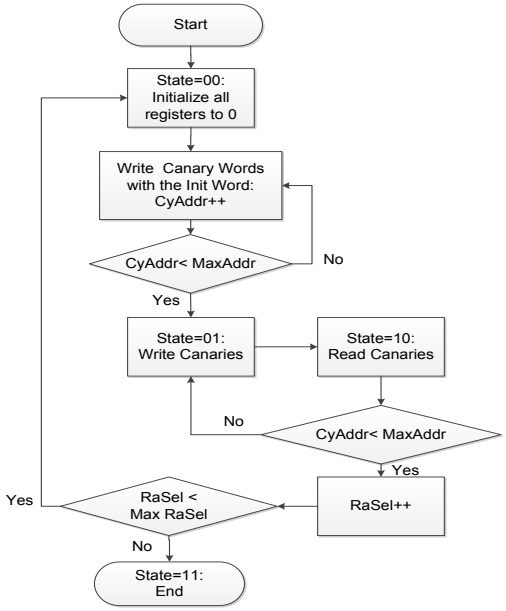


Fig 1: Algorithm for the Canary BIST Finite State Machine.

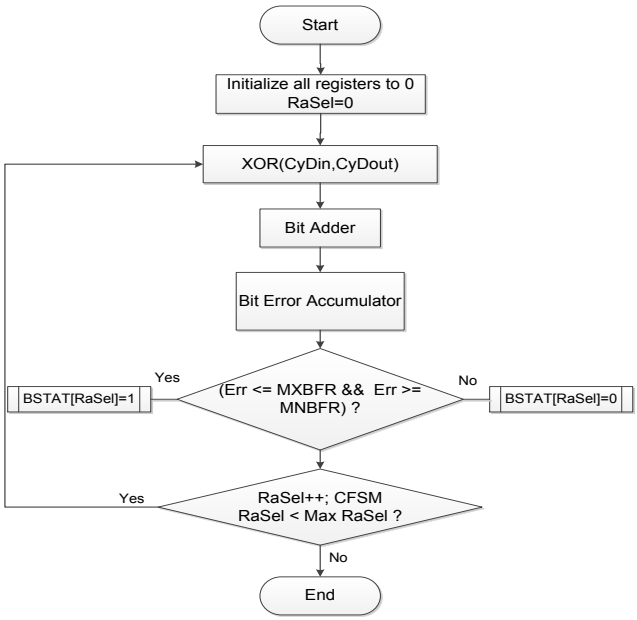


Fig 2: Algorithm for the Canary Bit Failure Rate FSM.

with the simulation- based verification, we do the synthesis using Synopsys's Design Compiler. Thereafter, we do the post-synthesis design verification using the VCS simulator using the same pre-synthesis test-bench. After the post-synthesis simulation based verification, we do DFT scan insertion in the CBIST using Synopsys's DFTMAX compiler . In parallel, we generate the Synopsys's TetraMAX compatible fault simulation files during the DFT insertion process. And lastly, we do the TetraMAX fault simulation of the CBIST itself assuming stuck at and transition fault models only for the test coverage and thus estimated the fault coverage.

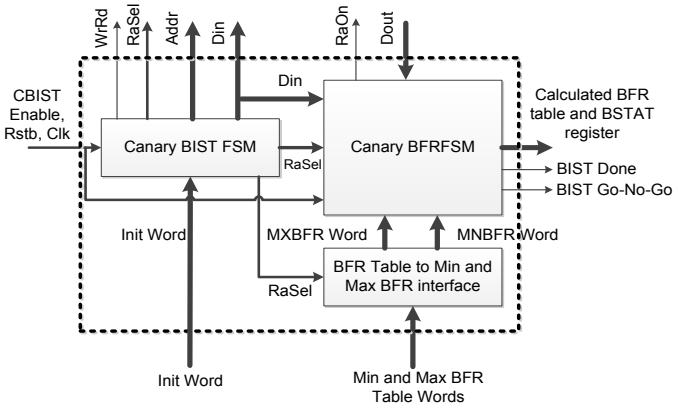
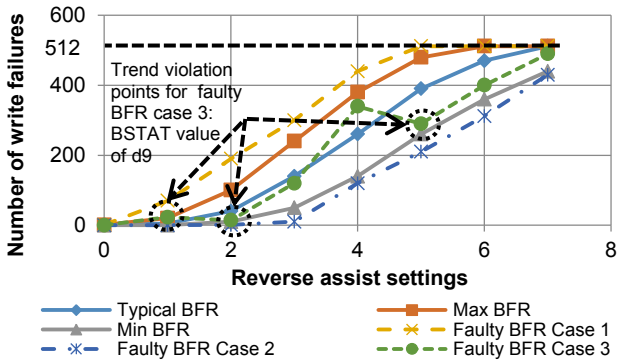


Fig 3: Architecture of the Canary BIST.



(a)

```
Calc BFR Signature is: 6e1684108c0c80a000006e1684108c0c80a00000
Read BFR Signature is:
6e1684108c0c80a000006e1684108c0c80a00000
BIST Go No Go Status is: 1
BIST Status Register Value is: ffff
```

(b)

```
Calc BFR Signature is: 7a990489541e00e05c007a990489541e00e05c00
Read BFR Signature is:
7a990489541e00e05c007a990489541e00e05c00
BIST Go No Go Status is: 0
BIST Status Register Value is: d9d9 (1101 1001 1101 1001)
```

(c)

Fig 4: (a) Write Bit Failure Rate vs. Reverse Assist Settings for various cases, (b) Post Synthesis Output for Minimum BFR Case, (c) Post Synthesis Output for Faulty BFR Case 3.

V. Results

Initially, we define the MAXBFR and MINBFR table data manually to create a bound for the calculated BFR during the RTL simulations. In order to see if the CBIST hardware works, we generate pre-determined total number of transition faults across the eight reverse assist settings manually and create six total cases to test the CBIST. The six cases includes the maximum, minimum and nominal BFR trends, a faulty BFR trend that overshoots the MXBFR trend (faulty BFR case 1), a faulty trend that undershoots the MNBFR trend (faulty BFR case 2) and faulty non-monotonic BFR trend (faulty BFR case 3) that has decreasing points with increasing RaSel value. We did the synthesis in SS_1.08V_-55C corner resulting in 36ns of cycle time or 27.77MHz operating frequency of the CBIST. We report that all six cases show correct outputs for the simulation of the synthesized netlist. Fig 4 (a) shows the number of write failure vs. RaSel for the six cases. Note that the BSTAT register value is 0XD9 (1101 1001) for the faulty BFR case 3 which matches exactly the same BFR locations as indicated in Fig 4 (a). Fig 4 (b) and (c) show the simulated signature comparison, CBIST status register value, and go-no-go status register value for the minimum BFR case and the faulty BFR case 3 that matches the computed BFR and overall status of the CBIST go-no-go output.

We report the estimated DFT test coverage of 99.6% for full-scan insertion in the CBIST. However, we have some issues with the DFT design rule checking (DRC) and due to that reason while running TetraMAX fault simulation using transition and stuck at fault models, we only got 64.74% and 74.31% test coverage respectively. Note that, the fault coverage is lower than those numbers in reality.

VI. Conclusion

The CBIST design shows promises to solve the issue of canary testing to see if we can use canary with reverse assist to track SRAM write V_{MIN}. This paper focuses on canary write failure trend checking only, but our hardware can be extended to readability testing also. The speed of the synthesized CBIST is not so good as the CBIST functionalities are heavy and we synthesized it using an older 130nm technology. Also, we face some issue in the scan insertion because of DFT DRC issues and that propagated to a below 99.6% estimated coverage. Whatsoever, overall, our project was successful and this project opens up new research directions in canary testing.

VII. Acknowledgement

We thank Dr. Benton H. Calhoun and the ECE7502S15 class students for providing valuable feedback and suggestions to make the project successful.

References

[1] R.W. Mann, et al. “Limits of Bias Based Assist Methods in Nano-Scale 6T SRAM”, in *Proc. ISQED*, pp. 1-8, 2010.

[2] A. Bhavnagarwala et al., “Fluctuation limits & scaling opportunities for CMOS SRAM cells,” in *IEDM Tech. Dig.*, 2005, pp. 659-662, 2005.

[3] J. Wang and B. Calhoun, “Canary replica feedback for near-DRV standby V_{DD} scaling in a 90 nm SRAM,” in *Proc. CICC '07*, pp. 29–32, 2007.

[4] A. Banerjee et al., “A reverse write assist circuit for SRAM dynamic write VMIN tracking using canary SRAMs,” in *ISQED, 2014*, pp.1,8, 3-5 March 2014.

[5] Kincel, A.; Balaz, M., "MBIST for LEON3 processor core cache," *DDECS, 2013*, pp.287,288, 8-10 April 2013.

.